<http://blog.csdn.net/gebitan505/article/details/44455235/>

使用SSM（Spring、SpringMVC和Mybatis）已经有三个多月了，项目在技术上已经没有什么难点了，基于现有的技术就可以实现想要的功能，当然肯定有很多可以改进的地方。之前没有记录SSM整合的过程，这次刚刚好基于自己的一个小项目重新搭建了一次，而且比项目搭建的要更好一些。以前解决问题的过程和方法并没有及时记录，以后在自己的小项目中遇到我再整理分享一下。这次，先说说三大框架整合过程。个人认为使用框架并不是很难，关键要理解其思想，这对于我们提高编程水平很有帮助。不过，如果用都不会，谈思想就变成纸上谈兵了！！！先技术，再思想。实践出真知。（可通过图片水印查看博客地址）

# 1、基本概念

## 1.1、Spring

        Spring是一个开源框架，Spring是于2003 年兴起的一个轻量级的Java 开发框架，由Rod Johnson 在其著作Expert One-On-One J2EE Development and Design中阐述的部分理念和原型衍生而来。它是为了解决企业应用开发的复杂性而创建的。Spring使用基本的JavaBean来完成以前只可能由EJB完成的事情。然而，Spring的用途不仅限于服务器端的开发。从简单性、可测试性和松耦合的角度而言，任何Java应用都可以从Spring中受益。 简单来说，Spring是一个轻量级的控制反转（IoC）和面向切面（AOP）的容器框架。

## 1.2、SpringMVC

        Spring MVC属于SpringFrameWork的后续产品，已经融合在Spring Web Flow里面。Spring MVC 分离了[控制器](http://baike.baidu.com/view/122229.htm" \t "_blank)、模型[对象](http://baike.baidu.com/view/2387.htm)、分派器以及处理程序对象的角色，这种分离让它们更容易进行定制。

## 1.3、MyBatis

       MyBatis 本是[apache](http://baike.baidu.com/view/28283.htm" \t "_blank)的一个开源项目[iBatis](http://baike.baidu.com/view/628102.htm" \t "_blank), 2010年这个项目由apache software foundation 迁移到了google code，并且改名为MyBatis 。MyBatis是一个基于Java的[持久层](http://baike.baidu.com/view/198047.htm" \t "_blank)框架。iBATIS提供的[持久层](http://baike.baidu.com/view/198047.htm" \t "_blank)框架包括SQL Maps和Data Access Objects（DAO）MyBatis 消除了几乎所有的JDBC代码和参数的手工设置以及结果集的检索。MyBatis 使用简单的 XML或注解用于配置和原始映射，将接口和 Java 的POJOs（Plain Old Java Objects，普通的 Java对象）映射成数据库中的记录。

# 2、开发环境搭建

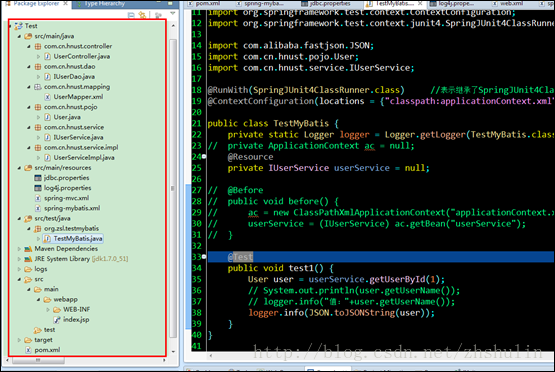
如果需要，参看之前的博文：<http://blog.csdn.net/zhshulin/article/details/30779873>

# 3、Maven Web项目创建

如果需要，参看之前的博文：<http://blog.csdn.net/zhshulin/article/details/37921705>

# 4、SSM整合

        下面主要介绍三大框架的整合，至于环境的搭建以及项目的创建，参看上面的博文。这次整合我分了2个配置文件，分别是spring-mybatis.xml，包含spring和mybatis的配置文件，还有个是spring-mvc的配置文件，此外有2个资源文件：jdbc.propertis和log4j.properties。完整目录结构如下（最后附上源码下载地址，不建议直接使用源码，因为此教程已经有了全部代码）：



使用框架都是较新的版本：

       Spring 4.0.2 RELEASE

       Spring MVC 4.0.2 RELEASE

       MyBatis 3.2.6

## 4.1、Maven引入需要的JAR包

         为了方便后面说的时候不需要引入JAR包，我这里直接给出所有需要的JAR包，这都是基本的JAR包，每个包的是干什么的都有注释，就不再多说了。

pom.xml

**[html]** [view plaincopy](http://blog.csdn.net/zhshulin/article/details/37956105)[在CODE上查看代码片](https://code.csdn.net/snippets/431711)

1. **<properties>**
2. <!-- spring版本号 -->
3. **<spring.version>**4.0.2.RELEASE**</spring.version>**
4. <!-- mybatis版本号 -->
5. **<mybatis.version>**3.2.6**</mybatis.version>**
6. <!-- log4j日志文件管理包版本 -->
7. **<slf4j.version>**1.7.7**</slf4j.version>**
8. **<log4j.version>**1.2.17**</log4j.version>**
9. **</properties>**
11. **<dependencies>**
12. **<dependency>**
13. **<groupId>**junit**</groupId>**
14. **<artifactId>**junit**</artifactId>**
15. **<version>**4.11**</version>**
16. <!-- 表示开发的时候引入，发布的时候不会加载此包 -->
17. **<scope>**test**</scope>**
18. **</dependency>**
19. <!-- spring核心包 -->
20. **<dependency>**
21. **<groupId>**org.springframework**</groupId>**
22. **<artifactId>**spring-core**</artifactId>**
23. **<version>**${spring.version}**</version>**
24. **</dependency>**
26. **<dependency>**
27. **<groupId>**org.springframework**</groupId>**
28. **<artifactId>**spring-web**</artifactId>**
29. **<version>**${spring.version}**</version>**
30. **</dependency>**
31. **<dependency>**
32. **<groupId>**org.springframework**</groupId>**
33. **<artifactId>**spring-oxm**</artifactId>**
34. **<version>**${spring.version}**</version>**
35. **</dependency>**
36. **<dependency>**
37. **<groupId>**org.springframework**</groupId>**
38. **<artifactId>**spring-tx**</artifactId>**
39. **<version>**${spring.version}**</version>**
40. **</dependency>**
42. **<dependency>**
43. **<groupId>**org.springframework**</groupId>**
44. **<artifactId>**spring-jdbc**</artifactId>**
45. **<version>**${spring.version}**</version>**
46. **</dependency>**
48. **<dependency>**
49. **<groupId>**org.springframework**</groupId>**
50. **<artifactId>**spring-webmvc**</artifactId>**
51. **<version>**${spring.version}**</version>**
52. **</dependency>**
53. **<dependency>**
54. **<groupId>**org.springframework**</groupId>**
55. **<artifactId>**spring-aop**</artifactId>**
56. **<version>**${spring.version}**</version>**
57. **</dependency>**
59. **<dependency>**
60. **<groupId>**org.springframework**</groupId>**
61. **<artifactId>**spring-context-support**</artifactId>**
62. **<version>**${spring.version}**</version>**
63. **</dependency>**
65. **<dependency>**
66. **<groupId>**org.springframework**</groupId>**
67. **<artifactId>**spring-test**</artifactId>**
68. **<version>**${spring.version}**</version>**
69. **</dependency>**
70. <!-- mybatis核心包 -->
71. **<dependency>**
72. **<groupId>**org.mybatis**</groupId>**
73. **<artifactId>**mybatis**</artifactId>**
74. **<version>**${mybatis.version}**</version>**
75. **</dependency>**
76. <!-- mybatis/spring包 -->
77. **<dependency>**
78. **<groupId>**org.mybatis**</groupId>**
79. **<artifactId>**mybatis-spring**</artifactId>**
80. **<version>**1.2.2**</version>**
81. **</dependency>**
82. <!-- 导入java ee jar 包 -->
83. **<dependency>**
84. **<groupId>**javax**</groupId>**
85. **<artifactId>**javaee-api**</artifactId>**
86. **<version>**7.0**</version>**
87. **</dependency>**
88. <!-- 导入Mysql数据库链接jar包 -->
89. **<dependency>**
90. **<groupId>**mysql**</groupId>**
91. **<artifactId>**mysql-connector-java**</artifactId>**
92. **<version>**5.1.30**</version>**
93. **</dependency>**
94. <!-- 导入dbcp的jar包，用来在applicationContext.xml中配置数据库 -->
95. **<dependency>**
96. **<groupId>**commons-dbcp**</groupId>**
97. **<artifactId>**commons-dbcp**</artifactId>**
98. **<version>**1.2.2**</version>**
99. **</dependency>**
100. <!-- JSTL标签类 -->
101. **<dependency>**
102. **<groupId>**jstl**</groupId>**
103. **<artifactId>**jstl**</artifactId>**
104. **<version>**1.2**</version>**
105. **</dependency>**
106. <!-- 日志文件管理包 -->
107. <!-- log start -->
108. **<dependency>**
109. **<groupId>**log4j**</groupId>**
110. **<artifactId>**log4j**</artifactId>**
111. **<version>**${log4j.version}**</version>**
112. **</dependency>**

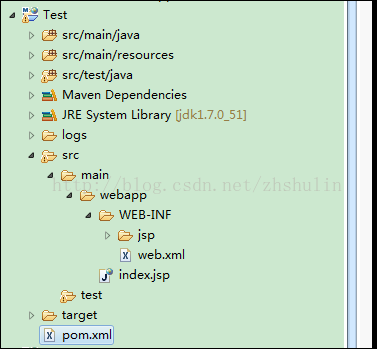
115. <!-- 格式化对象，方便输出日志 -->
116. **<dependency>**
117. **<groupId>**com.alibaba**</groupId>**
118. **<artifactId>**fastjson**</artifactId>**
119. **<version>**1.1.41**</version>**
120. **</dependency>**

123. **<dependency>**
124. **<groupId>**org.slf4j**</groupId>**
125. **<artifactId>**slf4j-api**</artifactId>**
126. **<version>**${slf4j.version}**</version>**
127. **</dependency>**
129. **<dependency>**
130. **<groupId>**org.slf4j**</groupId>**
131. **<artifactId>**slf4j-log4j12**</artifactId>**
132. **<version>**${slf4j.version}**</version>**
133. **</dependency>**
134. <!-- log end -->
135. <!-- 映入JSON -->
136. **<dependency>**
137. **<groupId>**org.codehaus.jackson**</groupId>**
138. **<artifactId>**jackson-mapper-asl**</artifactId>**
139. **<version>**1.9.13**</version>**
140. **</dependency>**
141. <!-- 上传组件包 -->
142. **<dependency>**
143. **<groupId>**commons-fileupload**</groupId>**
144. **<artifactId>**commons-fileupload**</artifactId>**
145. **<version>**1.3.1**</version>**
146. **</dependency>**
147. **<dependency>**
148. **<groupId>**commons-io**</groupId>**
149. **<artifactId>**commons-io**</artifactId>**
150. **<version>**2.4**</version>**
151. **</dependency>**
152. **<dependency>**
153. **<groupId>**commons-codec**</groupId>**
154. **<artifactId>**commons-codec**</artifactId>**
155. **<version>**1.9**</version>**
156. **</dependency>**

159. **</dependencies>**

## 4.2、Spring与MyBatis的整合

所有需要的JAR包都引入以后，首先进行Spring与MyBatis的整合，然后再进行JUnit测试，先看一个项目结构图：



### 4.2.1、建立JDBC属性文件

jdbc.properties（文件编码修改为utf-8）

**[html]** [view plaincopy](http://blog.csdn.net/zhshulin/article/details/37956105)[在CODE上查看代码片](https://code.csdn.net/snippets/431711)

1. driver=com.mysql.jdbc.Driver
2. url=jdbc:mysql://10.221.10.111:8080/db\_zsl
3. username=demao
4. password=demao
5. #定义初始连接数
6. initialSize=0
7. #定义最大连接数
8. maxActive=20
9. #定义最大空闲
10. maxIdle=20
11. #定义最小空闲
12. minIdle=1
13. #定义最长等待时间
14. maxWait=60000

### 4.2.2、建立spring-mybatis.xml配置文件

        这个文件就是用来完成spring和mybatis的整合的。这里面也没多少行配置，主要的就是自动扫描，自动注入，配置数据库。注释也很详细，大家看看就明白了。

spring-mybatis.xml

**[html]** [view plaincopy](http://blog.csdn.net/zhshulin/article/details/37956105)[在CODE上查看代码片](https://code.csdn.net/snippets/431711)

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<beans** xmlns="http://www.springframework.org/schema/beans"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:p="http://www.springframework.org/schema/p"
4. xmlns:context="http://www.springframework.org/schema/context"
5. xmlns:mvc="http://www.springframework.org/schema/mvc"
6. xsi:schemaLocation="http://www.springframework.org/schema/beans
7. http://www.springframework.org/schema/beans/spring-beans-3.1.xsd
8. http://www.springframework.org/schema/context
9. http://www.springframework.org/schema/context/spring-context-3.1.xsd
10. http://www.springframework.org/schema/mvc
11. http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd"**>**
12. <!-- 自动扫描 -->
13. **<context:component-scan** base-package="com.cn.hnust" **/>**
14. <!-- 引入配置文件 -->
15. **<bean** id="propertyConfigurer"
16. class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer"**>**
17. **<property** name="location" value="classpath:jdbc.properties" **/>**
18. **</bean>**
20. **<bean** id="dataSource" class="org.apache.commons.dbcp.BasicDataSource"
21. destroy-method="close"**>**
22. **<property** name="driverClassName" value="${driver}" **/>**
23. **<property** name="url" value="${url}" **/>**
24. **<property** name="username" value="${username}" **/>**
25. **<property** name="password" value="${password}" **/>**
26. <!-- 初始化连接大小 -->
27. **<property** name="initialSize" value="${initialSize}"**></property>**
28. <!-- 连接池最大数量 -->
29. **<property** name="maxActive" value="${maxActive}"**></property>**
30. <!-- 连接池最大空闲 -->
31. **<property** name="maxIdle" value="${maxIdle}"**></property>**
32. <!-- 连接池最小空闲 -->
33. **<property** name="minIdle" value="${minIdle}"**></property>**
34. <!-- 获取连接最大等待时间 -->
35. **<property** name="maxWait" value="${maxWait}"**></property>**
36. **</bean>**
38. <!-- spring和MyBatis完美整合，不需要mybatis的配置映射文件 -->
39. **<bean** id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean"**>**
40. **<property** name="dataSource" ref="dataSource" **/>**
41. <!-- 自动扫描mapping.xml文件 -->
42. **<property** name="mapperLocations" value="classpath:com/cn/hnust/mapping/\*.xml"**></property>**
43. **</bean>**
45. <!-- DAO接口所在包名，Spring会自动查找其下的类 -->
46. **<bean** class="org.mybatis.spring.mapper.MapperScannerConfigurer"**>**
47. **<property** name="basePackage" value="com.cn.hnust.dao" **/>**
48. **<property** name="sqlSessionFactoryBeanName" value="sqlSessionFactory"**></property>**
49. **</bean>**
51. <!-- (事务管理)transaction manager, use JtaTransactionManager for global tx -->
52. **<bean** id="transactionManager"
53. class="org.springframework.jdbc.datasource.DataSourceTransactionManager"**>**
54. **<property** name="dataSource" ref="dataSource" **/>**
55. **</bean>**
57. **</beans>**

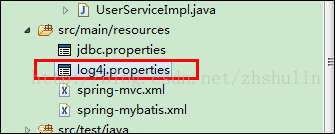
### 4.2.3、Log4j的配置

      为了方便调试，一般都会使用日志来输出信息，Log4j是[Apache](http://baike.baidu.com/subview/28283/5418753.htm" \t "_blank)的一个开放源代码项目，通过使用Log4j，我们可以控制[日志](http://baike.baidu.com/view/269351.htm" \t "_blank)信息输送的目的地是[控制台](http://baike.baidu.com/view/135167.htm)、文件、[GUI](http://baike.baidu.com/view/25309.htm)组件，甚至是套接口服务器、[NT](http://baike.baidu.com/subview/378/5113303.htm" \t "_blank)的事件记录器、[UNIX](http://baike.baidu.com/view/8095.htm" \t "_blank) [Syslog](http://baike.baidu.com/view/1614723.htm)[守护进程](http://baike.baidu.com/view/53123.htm)等；我们也可以控制每一条日志的输出格式；通过定义每一条日志信息的级别，我们能够更加细致地控制日志的生成过程。

      Log4j的配置很简单，而且也是通用的，下面给出一个基本的配置，换到其他项目中也无需做多大的调整，如果想做调整或者想了解Log4j的各种配置，参看我转载的一篇博文，很详细：

<http://blog.csdn.net/zhshulin/article/details/37937365>

下面给出配置文件目录：



log4j.properties

**[html]** [view plaincopy](http://blog.csdn.net/zhshulin/article/details/37956105)[在CODE上查看代码片](https://code.csdn.net/snippets/431711)

1. #定义LOG输出级别
2. log4j.rootLogger=INFO,Console,File
3. #定义日志输出目的地为控制台
4. log4j.appender.Console=org.apache.log4j.ConsoleAppender
5. log4j.appender.Console.Target=System.out
6. #可以灵活地指定日志输出格式，下面一行是指定具体的格式
7. log4j.appender.Console.layout = org.apache.log4j.PatternLayout
8. log4j.appender.Console.layout.ConversionPattern=[%c] - %m%n
10. #文件大小到达指定尺寸的时候产生一个新的文件
11. log4j.appender.File = org.apache.log4j.RollingFileAppender
12. #指定输出目录
13. log4j.appender.File.File = logs/ssm.log
14. #定义文件最大大小
15. log4j.appender.File.MaxFileSize = 10MB
16. # 输出所以日志，如果换成DEBUG表示输出DEBUG以上级别日志
17. log4j.appender.File.Threshold = ALL
18. log4j.appender.File.layout = org.apache.log4j.PatternLayout
19. log4j.appender.File.layout.ConversionPattern =[%p] [%d{yyyy-MM-dd HH\:mm\:ss}][%c]%m%n

### 4.2.4、JUnit测试

      经过以上步骤（到4.2.2，log4j不配也没影响），我们已经完成了Spring和mybatis的整合，这样我们就可以编写一段测试代码来试试是否成功了。

#### 4.2.4.1、创建测试用表

既然我们需要测试，那么我们就需要建立在数据库中建立一个测试表，这个表建的很简单，SQL语句为：

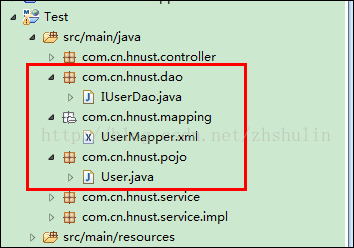
**[sql]** [view plaincopy](http://blog.csdn.net/zhshulin/article/details/37956105)[在CODE上查看代码片](https://code.csdn.net/snippets/431711)

1. **DROP** **TABLE** IF EXISTS `user\_t`;
3. **CREATE** **TABLE** `user\_t` (
4. `id` **int**(11) NOT NULL AUTO\_INCREMENT,
5. `user\_name` **varchar**(40) NOT NULL,
6. `**password**` **varchar**(255) NOT NULL,
7. `age` **int**(4) NOT NULL,
8. **PRIMARY** **KEY** (`id`)
9. ) ENGINE=InnoDB AUTO\_INCREMENT=2 **DEFAULT** CHARSET=utf8;
11. /\*Data **for** the **table** `user\_t` \*/
13. **insert**  **into** `user\_t`(`id`,`user\_name`,`**password**`,`age`) **values** (1,'测试','sfasgfaf',24);

#### 4.2.4.2、利用MyBatis Generator自动创建代码

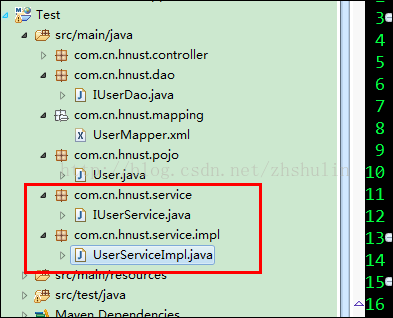
参考博文：<http://blog.csdn.net/zhshulin/article/details/23912615>

       这个可根据表自动创建实体类、MyBatis映射文件以及DAO接口，当然，我习惯将生成的接口名改为IUserDao，而不是直接用它生成的UserMapper。如果不想麻烦就可以不改。完成后将文件复制到工程中。如图：



#### 4.2.4.3、建立Service接口和实现类

目录结构：



下面给出具体的内容：

IUserService.jave

**[java]** [view plaincopy](http://blog.csdn.net/zhshulin/article/details/37956105)[在CODE上查看代码片](https://code.csdn.net/snippets/431711)

1. **package** com.cn.hnust.service;
3. **import** com.cn.hnust.pojo.User;
5. **public** **interface** IUserService {
6. **public** User getUserById(**int** userId);
7. }

UserServiceImpl.java

**[java]** [view plaincopy](http://blog.csdn.net/zhshulin/article/details/37956105)[在CODE上查看代码片](https://code.csdn.net/snippets/431711)

1. **package** com.cn.hnust.service.impl;
3. **import** javax.annotation.Resource;
5. **import** org.springframework.stereotype.Service;
7. **import** com.cn.hnust.dao.IUserDao;
8. **import** com.cn.hnust.pojo.User;
9. **import** com.cn.hnust.service.IUserService;
11. @Service("userService")
12. **public** **class** UserServiceImpl **implements** IUserService {
13. @Resource
14. **private** IUserDao userDao;
15. @Override
16. **public** User getUserById(**int** userId) {
17. // TODO Auto-generated method stub
18. **return** **this**.userDao.selectByPrimaryKey(userId);
19. }
21. }

#### 4.2.4.4、建立测试类

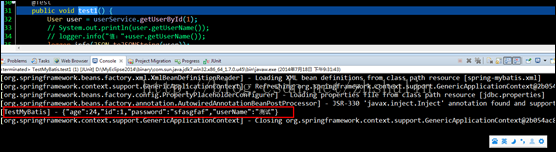
       测试类在src/test/java中建立，下面测试类中注释掉的部分是不使用Spring时，一般情况下的一种测试方法；如果使用了Spring那么就可以使用注解的方式来引入配置文件和类，然后再将service接口对象注入，就可以进行测试了。

       如果测试成功，表示Spring和Mybatis已经整合成功了。输出信息使用的是Log4j打印到控制台。

**[java]** [view plaincopy](http://blog.csdn.net/zhshulin/article/details/37956105)[在CODE上查看代码片](https://code.csdn.net/snippets/431711)

1. **package** org.zsl.testmybatis;
3. **import** javax.annotation.Resource;
5. **import** org.apache.log4j.Logger;
6. **import** org.junit.Before;
7. **import** org.junit.Test;
8. **import** org.junit.runner.RunWith;
9. **import** org.springframework.context.ApplicationContext;
10. **import** org.springframework.context.support.ClassPathXmlApplicationContext;
11. **import** org.springframework.test.context.ContextConfiguration;
12. **import** org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
14. **import** com.alibaba.fastjson.JSON;
15. **import** com.cn.hnust.pojo.User;
16. **import** com.cn.hnust.service.IUserService;
18. @RunWith(SpringJUnit4ClassRunner.**class**)     //表示继承了SpringJUnit4ClassRunner类
19. @ContextConfiguration(locations = {"classpath:spring-mybatis.xml"})
21. **public** **class** TestMyBatis {
22. **private** **static** Logger logger = Logger.getLogger(TestMyBatis.**class**);
23. //  private ApplicationContext ac = null;
24. @Resource
25. **private** IUserService userService = **null**;
27. //  @Before
28. //  public void before() {
29. //      ac = new ClassPathXmlApplicationContext("applicationContext.xml");
30. //      userService = (IUserService) ac.getBean("userService");
31. //  }
33. @Test
34. **public** **void** test1() {
35. User user = userService.getUserById(1);
36. // System.out.println(user.getUserName());
37. // logger.info("值："+user.getUserName());
38. logger.info(JSON.toJSONString(user));
39. }
40. }

测试结果：



至此，完成Spring和mybatis这两大框架的整合，下面在继续进行SpringMVC的整合。

## 4.3、整合SpringMVC

         上面已经完成了2大框架的整合，SpringMVC的配置文件单独放，然后在web.xml中配置整合。

### 4.3.1、配置spring-mvc.xml

配置里面的注释也很详细，在此就不说了，主要是自动扫描控制器，视图模式，注解的启动这三个。

**[html]** [view plaincopy](http://blog.csdn.net/zhshulin/article/details/37956105)[在CODE上查看代码片](https://code.csdn.net/snippets/431711)

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<beans** xmlns="http://www.springframework.org/schema/beans"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:p="http://www.springframework.org/schema/p"
4. xmlns:context="http://www.springframework.org/schema/context"
5. xmlns:mvc="http://www.springframework.org/schema/mvc"
6. xsi:schemaLocation="http://www.springframework.org/schema/beans
7. http://www.springframework.org/schema/beans/spring-beans-3.1.xsd
8. http://www.springframework.org/schema/context
9. http://www.springframework.org/schema/context/spring-context-3.1.xsd
10. http://www.springframework.org/schema/mvc
11. http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd"**>**
12. <!-- 自动扫描该包，使SpringMVC认为包下用了@controller注解的类是控制器 -->
13. **<context:component-scan** base-package="com.cn.hnust.controller" **/>**
14. <!--避免IE执行AJAX时，返回JSON出现下载文件 -->
15. **<bean** id="mappingJacksonHttpMessageConverter"
16. class="org.springframework.http.converter.json.MappingJacksonHttpMessageConverter"**>**
17. **<property** name="supportedMediaTypes"**>**
18. **<list>**
19. **<value>**text/html;charset=UTF-8**</value>**
20. **</list>**
21. **</property>**
22. **</bean>**
23. <!-- 启动SpringMVC的注解功能，完成请求和注解POJO的映射 -->
24. **<bean**
25. class="org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter"**>**
26. **<property** name="messageConverters"**>**
27. **<list>**
28. **<ref** bean="mappingJacksonHttpMessageConverter" **/>** <!-- JSON转换器 -->
29. **</list>**
30. **</property>**
31. **</bean>**
32. <!-- 定义跳转的文件的前后缀 ，视图模式配置-->
33. **<bean** class="org.springframework.web.servlet.view.InternalResourceViewResolver"**>**
34. <!-- 这里的配置我的理解是自动给后面action的方法return的字符串加上前缀和后缀，变成一个 可用的url地址 -->
35. **<property** name="prefix" value="/WEB-INF/jsp/" **/>**
36. **<property** name="suffix" value=".jsp" **/>**
37. **</bean>**
39. <!-- 配置文件上传，如果没有使用文件上传可以不用配置，当然如果不配，那么配置文件中也不必引入上传组件包 -->
40. **<bean** id="multipartResolver"
41. class="org.springframework.web.multipart.commons.CommonsMultipartResolver"**>**
42. <!-- 默认编码 -->
43. **<property** name="defaultEncoding" value="utf-8" **/>**
44. <!-- 文件大小最大值 -->
45. **<property** name="maxUploadSize" value="10485760000" **/>**
46. <!-- 内存中的最大值 -->
47. **<property** name="maxInMemorySize" value="40960" **/>**
48. **</bean>**
50. **</beans>**

### 4.3.2、配置web.xml文件

         这里面对spring-mybatis.xml的引入以及配置的spring-mvc的Servlet就是为了完成SSM整合，之前2框架整合不需要在此处进行任何配置。配置一样有详细注释，不多解释了。

web.xml

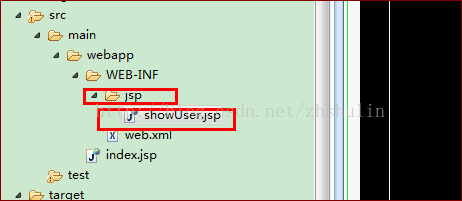
**[html]** [view plaincopy](http://blog.csdn.net/zhshulin/article/details/37956105)[在CODE上查看代码片](https://code.csdn.net/snippets/431711)

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<web-app** xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3. xmlns="http://java.sun.com/xml/ns/javaee"
4. xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"
5. version="3.0"**>**
6. **<display-name>**Archetype Created Web Application**</display-name>**
7. <!-- Spring和mybatis的配置文件 -->
8. **<context-param>**
9. **<param-name>**contextConfigLocation**</param-name>**
10. **<param-value>**classpath:spring-mybatis.xml**</param-value>**
11. **</context-param>**
12. <!-- 编码过滤器 -->
13. **<filter>**
14. **<filter-name>**encodingFilter**</filter-name>**
15. **<filter-class>**org.springframework.web.filter.CharacterEncodingFilter**</filter-class>**
16. **<async-supported>**true**</async-supported>**
17. **<init-param>**
18. **<param-name>**encoding**</param-name>**
19. **<param-value>**UTF-8**</param-value>**
20. **</init-param>**
21. **</filter>**
22. **<filter-mapping>**
23. **<filter-name>**encodingFilter**</filter-name>**
24. **<url-pattern>**/\***</url-pattern>**
25. **</filter-mapping>**
26. <!-- Spring监听器 -->
27. **<listener>**
28. **<listener-class>**org.springframework.web.context.ContextLoaderListener**</listener-class>**
29. **</listener>**
30. <!-- 防止Spring内存溢出监听器 -->
31. **<listener>**
32. **<listener-class>**org.springframework.web.util.IntrospectorCleanupListener**</listener-class>**
33. **</listener>**
35. <!-- Spring MVC servlet -->
36. **<servlet>**
37. **<servlet-name>**SpringMVC**</servlet-name>**
38. **<servlet-class>**org.springframework.web.servlet.DispatcherServlet**</servlet-class>**
39. **<init-param>**
40. **<param-name>**contextConfigLocation**</param-name>**
41. **<param-value>**classpath:spring-mvc.xml**</param-value>**
42. **</init-param>**
43. **<load-on-startup>**1**</load-on-startup>**
44. **<async-supported>**true**</async-supported>**
45. **</servlet>**
46. **<servlet-mapping>**
47. **<servlet-name>**SpringMVC**</servlet-name>**
48. <!-- 此处可以可以配置成\*.do，对应struts的后缀习惯 -->
49. **<url-pattern>**/**</url-pattern>**
50. **</servlet-mapping>**
51. **<welcome-file-list>**
52. **<welcome-file>**/index.jsp**</welcome-file>**
53. **</welcome-file-list>**
55. **</web-app>**

### 4.3.3、测试

        至此已经完成了SSM三大框架的整合了，接下来测试一下，如果成功了，那么恭喜你，如果失败了，继续调试吧，作为程序员就是不停的与BUG做斗争！

#### 4.3.3.1、新建jsp页面



showUser.jsp   此页面仅输出一下用户名，完成一个完整的简单流程。

**[html]** [view plaincopy](http://blog.csdn.net/zhshulin/article/details/37956105)[在CODE上查看代码片](https://code.csdn.net/snippets/431711)

1. **<**%@ page language="java" import="java.util.\*" pageEncoding="utf-8"%**>**
2. <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"**>**
3. **<html>**
4. **<head>**
5. **<title>**测试**</title>**
6. **</head>**
8. **<body>**
9. ${user.userName}
10. **</body>**
11. **</html>**

#### 4.3.3.2、建立UserController类

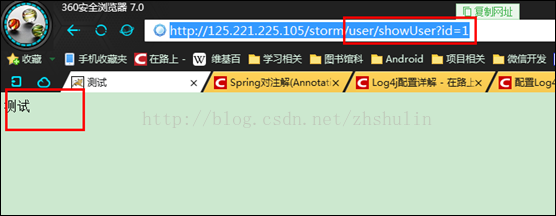
UserController.java  控制器

**[java]** [view plaincopy](http://blog.csdn.net/zhshulin/article/details/37956105)[在CODE上查看代码片](https://code.csdn.net/snippets/431711)

1. **package** com.cn.hnust.controller;
3. **import** javax.annotation.Resource;
4. **import** javax.servlet.http.HttpServletRequest;
6. **import** org.springframework.stereotype.Controller;
7. **import** org.springframework.ui.Model;
8. **import** org.springframework.web.bind.annotation.RequestMapping;
10. **import** com.cn.hnust.pojo.User;
11. **import** com.cn.hnust.service.IUserService;
13. @Controller
14. @RequestMapping("/user")
15. **public** **class** UserController {
16. @Resource
17. **private** IUserService userService;
19. @RequestMapping("/showUser")
20. **public** String toIndex(HttpServletRequest request,Model model){
21. **int** userId = Integer.parseInt(request.getParameter("id"));
22. User user = **this**.userService.getUserById(userId);
23. model.addAttribute("user", user);
24. **return** "showUser";
25. }
26. }

#### 4.3.3.3、部署项目

输入地址：localhost:8080/项目名称/user/showUser?id=1



至此，SSM三大框架的整合就完成了，在此基础上可再添加其他功能。

源码下载地址：<http://download.csdn.net/detail/u012909091/7658611>

 （转载注明出处：<http://blog.csdn.net/zhshulin>）